CSE 2336 Assignment 4

NOTES:

Each program should include comments that explain what each block of code is doing. Addi6onally, the programs should compile without errors, and run with the results described in the exercise. The following deduc6ons will be made from each exercise if any of the following is incorrect or missing:

Proper formatting [5 points]
Proper names for classes and variables [5 points]
Comments [5 point]
Program doesn't compile [10 points]
Source code (java file) missing [10 points]
Executable (class file) missing [10 points]
Missing array where an array was required [5 points]
Missing loop where a loop was required [5 points]
Missing class from the design provided [10 points]
Missing method from the design provided [5 points[

This Lab is due Saturday November 10 at 6:00am.

Lab (100 Points)

Create a Java for the game Dallopoly. The game board is depicted on the next page. The

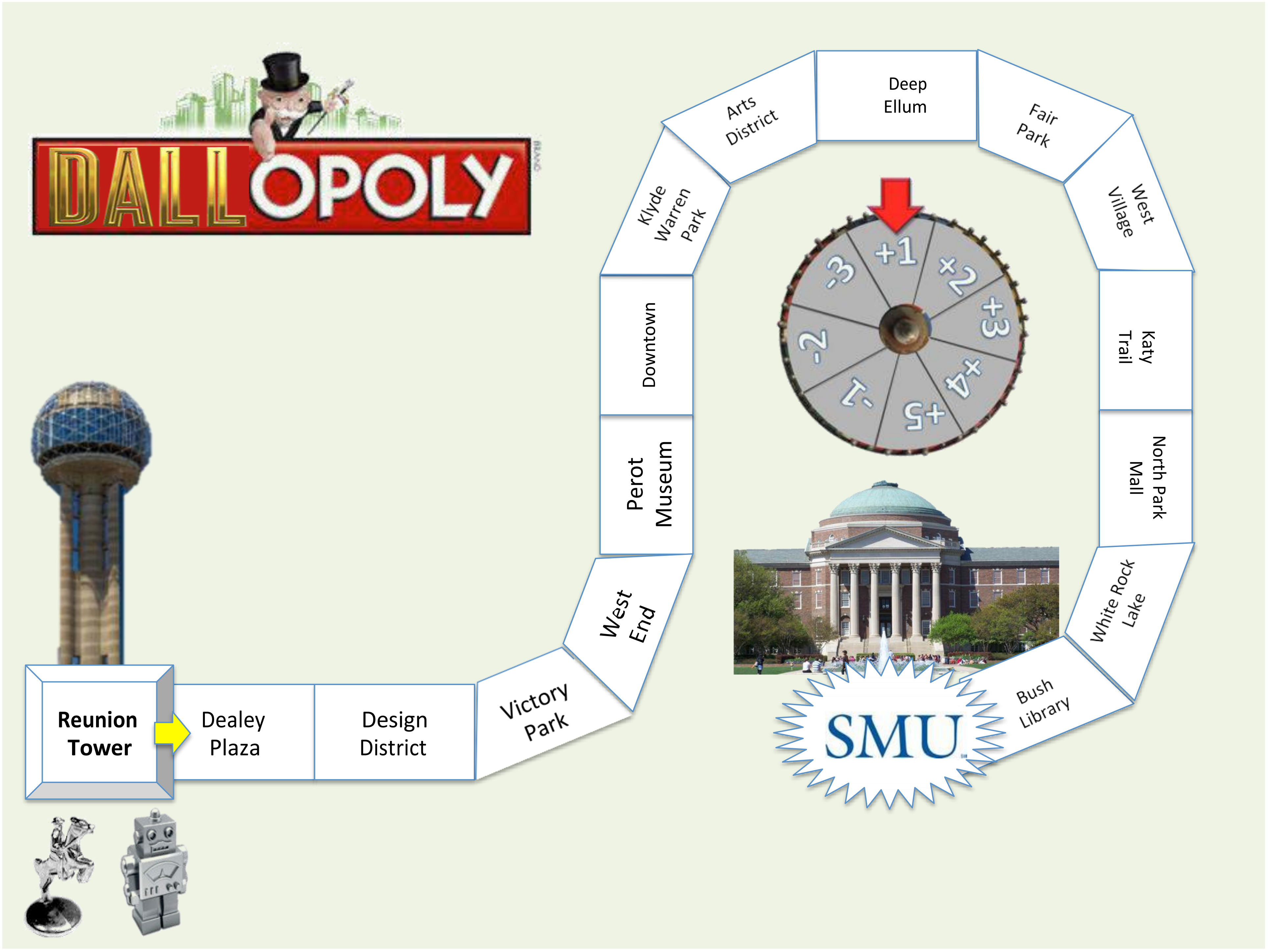
objective of the game is to start on the Reunion Tower square and get to SMU.

Each player takes a turn by spinning a wheel that contains the values 1, 2, 3, 4, 5, 11, 12, and 13. The player will move ahead (or back, if a nega6ve number) the number of squares on the board indicated by the spin value. If the spin value causes a player to move beyond the end of the board or ahead of the startng square, the player may not move. In that case, the player will remain on the same square and play resumes with the next player.

The game ends when the first player lands directly on the SMU square.

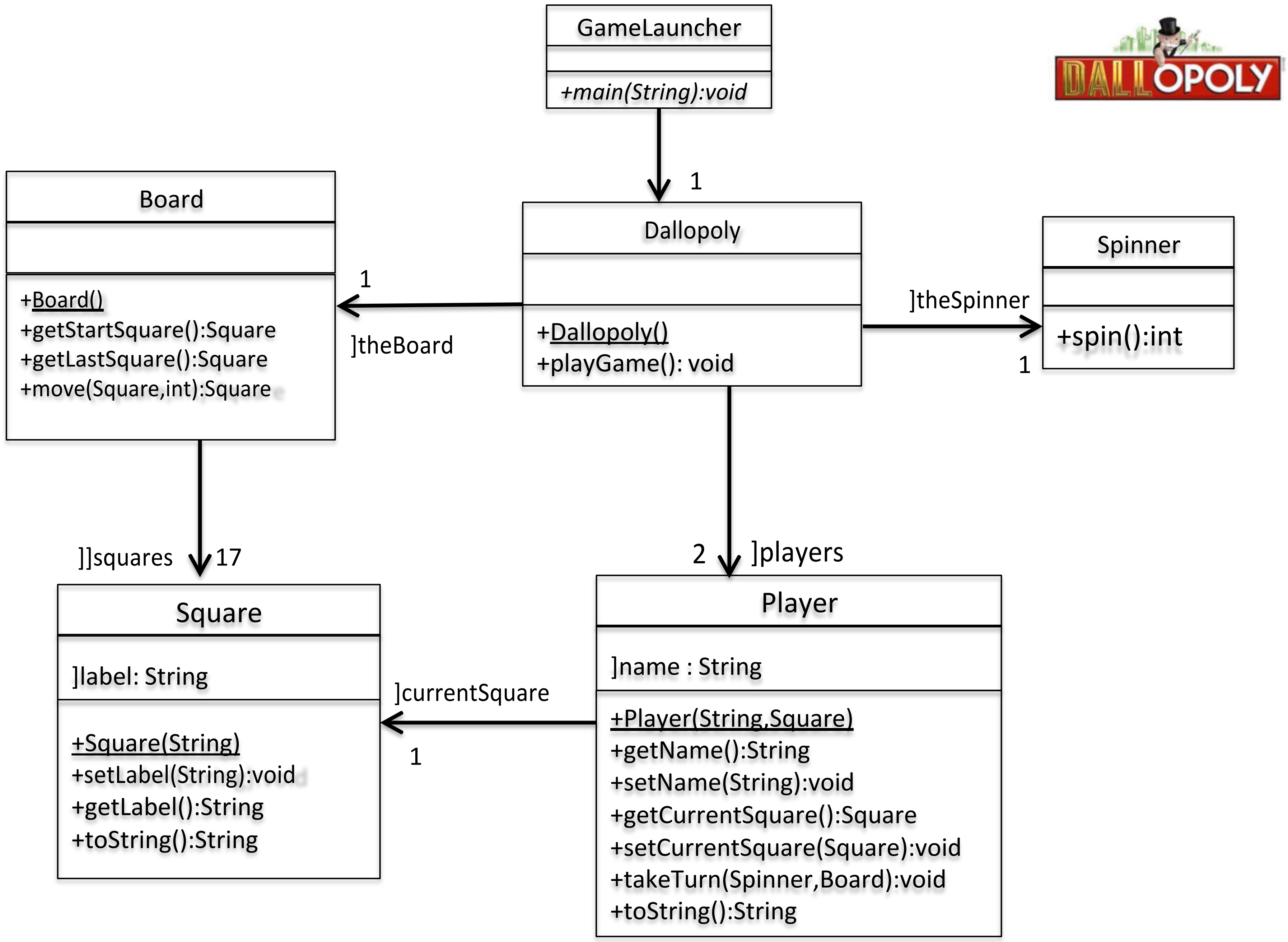
A UML design is provided that you must follow when building this solu6on. Read the instruc6ons and 6ps provided on the model very carefully. Each class in the model is described in detail on separate pages in the instruc6ons.

When you run the program, there will be no user interac6on. This is a simulation of two players playing the game from start to finish. Your output should depict each step of the game with the spin value and board location of each player after each turn. Sample output is provided in the instruc6ons.

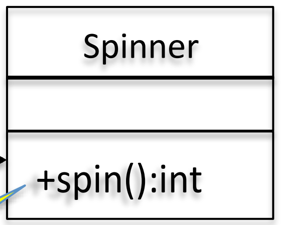

DALLOPOLY
Reunion Tower
Dealey Plaza
Design District
Victory Park
West End
Perot Museum
Downtown
Klyde Warren Park
Arts District
Deep Ellum
Fair Park
West Village
Katy Trail
North Park Mall
White Rock Lake
Bush Library
SMU
+1
+2
+3
+4
+5
-1
-2
-3

Sample Output:

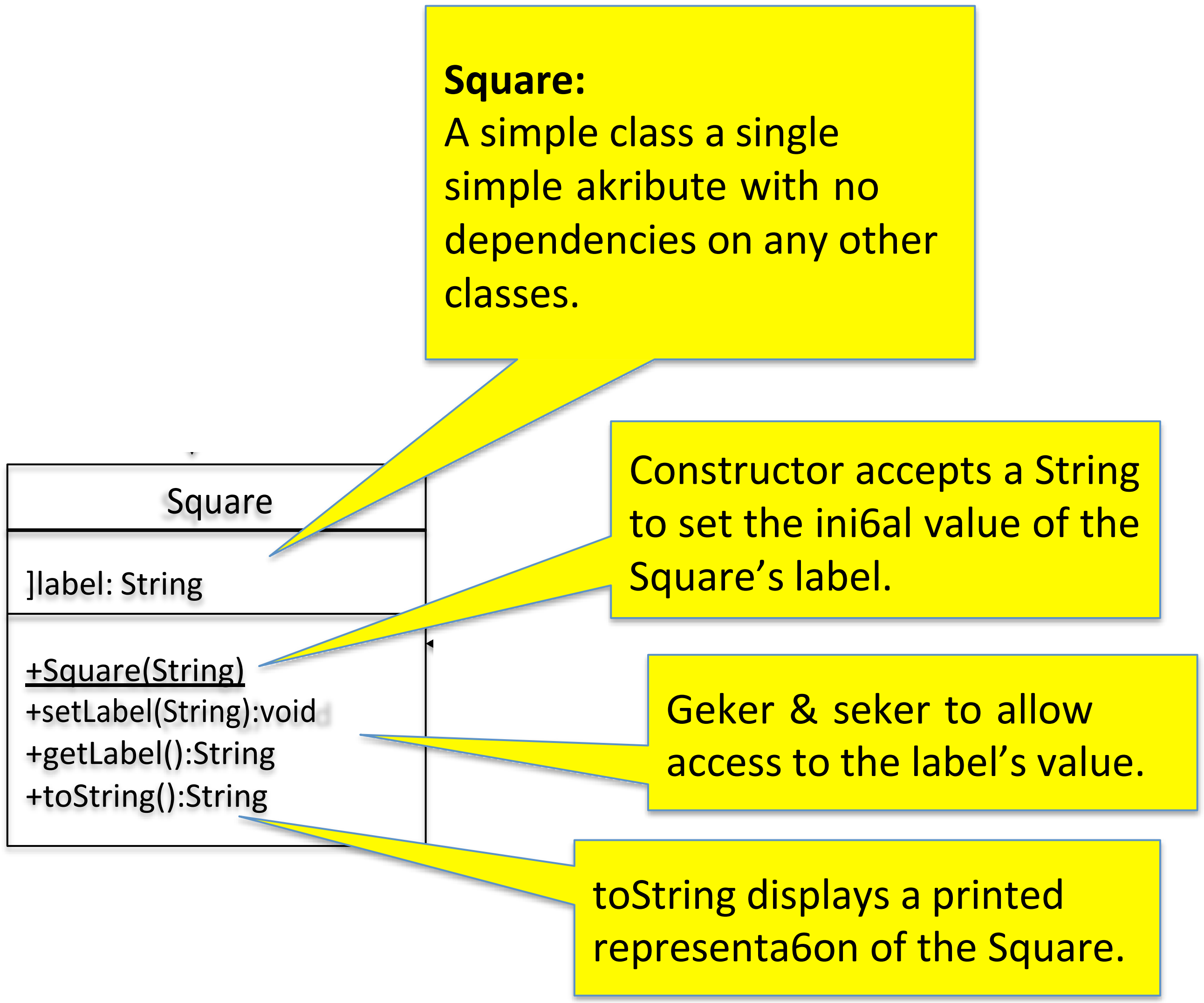
```
> java GameLauncher
Star6ng the game...

Horse is on Reunion Tower square
Robot is on Reunion Tower square

Horse spun 2 Horse is on Design District square
Robot spun 3 Robot is on Victory Park square

Horse spun 5 Horse is on Klyde Warren Park square
Robot spun 3 Robot is on Downtown square

Horse spun 3 Horse is on Fair Park square
Robot spun 5 Robot is on West Village square

Horse spun ]3 Horse is on Klyde Warren Park square
Robot spun 3 Robot is on White Rock Lake square

Horse spun 4 Horse is on West Village square
Robot spun ]2 Robot is on Katy Trail square

Horse spun 3 Horse is on White Rock Lake square
Robot spun ]3 Robot is on Deep Ellum square

Horse spun 3 Horse is on White Rock Lake square
Robot spun ]3 Robot is on Downtown square

Horse spun 2 Horse is on SMU square
GAME OVER!!!  WINNER HORSE IS ON SMU SQUARE
```

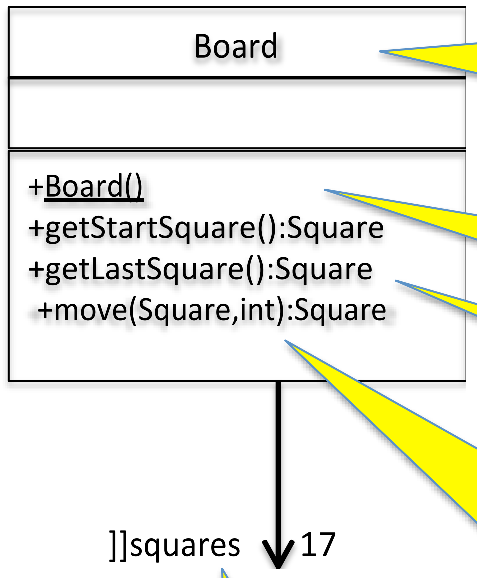

GameLauncher
+main(String):void
1
DALLOPOLY
Board
+Board()
+getStartSquare():Square
+getLastSquare():Square
+move(Square,int):Square
1
]theBoard
Dallopoly
+Dallopoly()
+playGame(): void
]theSpinner
Spinner
+spin():int
1
]]squares
17
2
]players
Square
]label: String
+Square(String)
+setLabel(String):void
+getLabel():String
+toString():String
]currentSquare
1
Player
]name : String
+Player(String,Square)
+getName():String
+setName(String):void
+getCurrentSquare():Square
+setCurrentSquare(Square):void
+takeTurn(Spinner,Board):void
+toString():String

Spinner:
A simple class with no akributes and no dependencies on any other classes.



spin() method returns a random selec6on of the values 1,2,3,4,5,]1,]2, or]3


Square
]label: String
+Square(String)
+setLabel(String):void
+getLabel():String
+toString():String
Square:
A simple class a single simple akribute with no dependencies on any other classes.
Constructor accepts a String to set the ini6al value of the Square's label.
Geker & seker to allow access to the label's value.
toString displays a printed representa6on of the Square.



Board:
Creates and holds onto all of the 17 Square objects. No simple akributes and one reference akribute.

Constructor creates the ArrayList *squares* and creates each of the 17 Square objects (with unique names – see game board for names) and retains them in that ArrayList.

getStartSquare() returns the Square at the first spot in the ArrayList.
getLastSquare() returns the Square in the last spot of the ArrayList

squares reference akribute is an ArrayList containing the 17 Square objects.

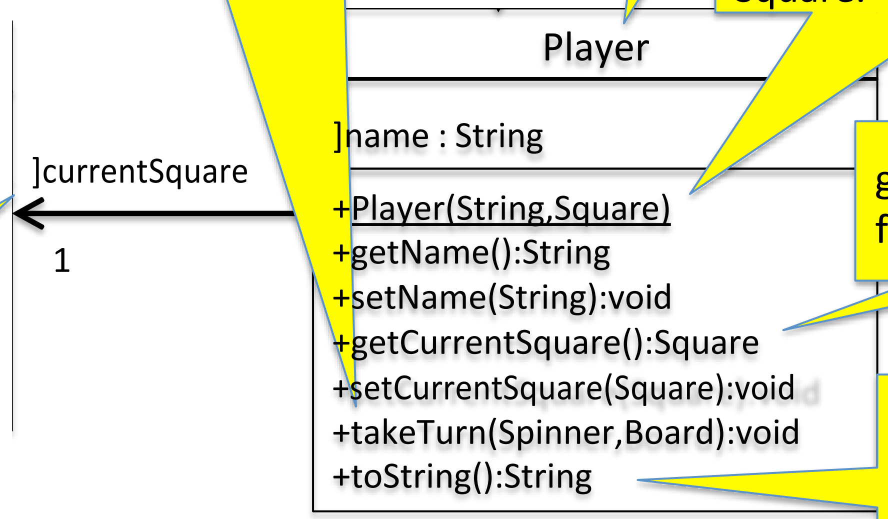
move() method accepts two parameters: a Square and an int. Determine the Square that is that number of spots away on the board and return it. If the *int* value would cause the player to move off the end (or off the beginning) of the board, just return the Square that was passed in because the Player cannot move if the spin value exceeds the size of the board.

takeTurn accepts the game's Spinner and Board to allow the Player to communicate with those objects. Sends the *spin* message to the Spinner, which will return the spin value. Then sends the spin value and the current square to the board's *move* method. The board will return the Player's new square loca6on. Update the value of currentSquare with the returned Square.

Player:
Has one simple akribute (name) and one reference akribute which holds a reference to the Square the Player is currently silng on.

The constructor accepts two parameters: A String containing the Player's name, and the Board's star6ng Square.

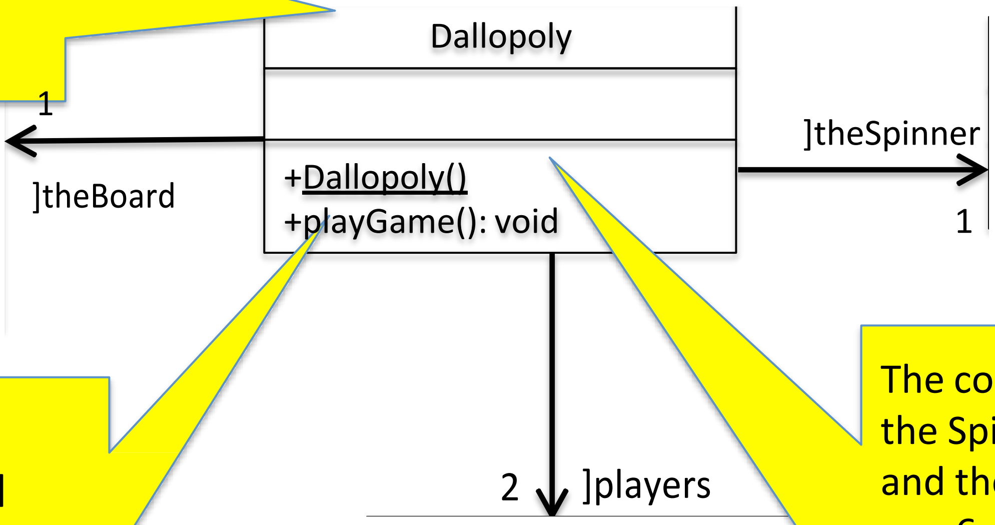
gekers and sekers for both akributes.



reference akribute containing Player's loca6on on the board.

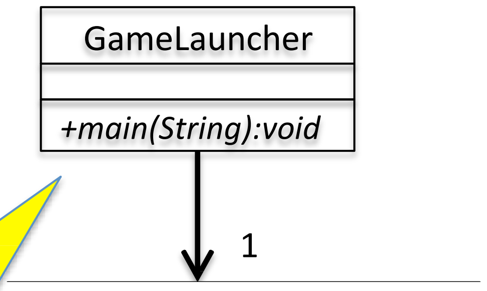
toString returns Player's name and current Square .

Dallopoly:
The central game class. The Constructor



playGame() method:
Outer loop repeats un6l one of the player wins by landing on the last square. Inner loop alternates players so each gets a turn.

The constructor creates the Spinner, the Board, and the Players. When crea6ng each Player, pass the name and a reference to the board's star6ng square. (which you must first get from the board.) Print each Player to the console a@er crea6ng them and adding them to the players ArrayList.



GameLauncher: Board

```
Dallopoly game = new Dallopoly();
game.playGame();
```

```
public void playGame()
    {
    //seet a flag that will change when the game ends
    boolean gameOver = false;

    //Create an outer loop that exits when a player wins
    while(gameOver == false)
    {

    //Loop through the ArrayList of players and
    //send each player the "takeTurn" message.
    //After the player has moved, compare the player’s new square
    //to the Board’s lastSquare.  If the player is on the Board’s
    //last square, declare that player the winner and exit the
    //while loop.


       //INSERT MISSING CODE HERE


    }//end while
    }//end playGame
```